WHAT IS CLAIMED IS:

Sub A2

1. A method for renaming memory references to stack locations in a computer processing system, comprising the steps of:

detecting stack references that use architecturally defined stack access methods; and

replacing the stack references with references to processor-internal registers.

2. The method according to claim 1, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.

3. The method according to claim 2, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

4. The method according to claim 1, further comprising the step of performing a consistency-preserving

Cont A2

operation for a stack reference that does not use the architecturally defined stack access methods.

5. The method according to claim 4, wherein said step of performing a consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that references a stack area and that does not use the architecturally defined stack access methods.

6. The method according to claim 4, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing a consistency-preserving operation comprises the step of recovering an in-order value for the stack reference from the main memory, upon performing said synchronizing step.

7. The method according claim 6, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an

instruction corresponding to the stack reference, when the stack reference is a write stack reference.

8. The method according to claim 6, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

9. The method according to claim 4, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state of a processor the system;

synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.

10. The method according to claim 1, wherein the architecturally defined stack access methods comprise memory

accesses that use at least one of a stack pointer, a frame pointer, and an argument pointer.

A2

11. The method according to claim 1, wherein the architecturally defined stack access methods comprise push, pop, and other stack manipulation operations.

12. A method for renaming memory references to stack locations in a computer processing system, comprising the steps of:

determining whether a load instruction references a location in a local stack using an architecturally defined register for accessing a stack location;

determining whether a rename register exists for the referenced location in the local stack, when the load instruction references the location using the architecturally defined register; and

replacing the reference to the location by a reference to the rename register, when the rename register exists.

13. The method according to claim 12, wherein the architecturally defined register corresponds to a pointer for accessing stack locations.

A2

14. The method according to claim 13, wherein the pointer for accessing the stack locations is one of a stack pointer, a frame pointer, and an argument pointer.

15. The method according to claim 12, wherein the architecturally defined stack access methods comprise push, pop, and other stack manipulation operations.

16. The method according to claim 12, wherein said step of determining whether the renaming register exists comprises the step of computing one of a symbolic address and an actual address of the location.

17. The method according to claim 12, wherein said step of determining whether the rename register exists is performed during one of a decode, an address generation, and a memory access phase.

18. The method according to claim 12, further comprising the step of performing the load instruction from one of a main memory and a cache of the system, when the rename register does not exist.

A2

19. The method according to claim 12, further comprising the step of determining whether the load instruction references a location in any stack, including the local stack, using another register, when the load instruction does not reference the location using the architecturally defined register.

20. The method according to claim 19, wherein said step of determining whether the load instruction references the location using the other register comprises the step of marking translation lookaside buffer (TLB) entries of pages in the local stack as containing stack references, when the load instruction references the location using the other register.

21. The method according to claim 19, further comprising the step of performing the load instruction from one of a main memory and a cache of the system, when the load instruction does not reference the location using the other register.

A2

22. The method according to claim 19, further comprising the step of executing a consistency-preserving mechanism to perform the load instruction from the stack area, when the load instruction references the location using the other register.

23. The method according to claim 12, further comprising the step of allocating a rename register for the location, when the rename register does not exist.

24. The method according to claim 23, further comprising the step of inserting an operation, into an instruction stream containing the load instruction, to load the location from a processor of the system to the allocated rename register, upon allocating the rename register.

25. The method according to claim 24, further comprising the step of replacing the reference to the location by a reference to the allocated rename register, upon inserting the operation.

A2

26. A method for renaming memory references to stack locations in a computer processing system, comprising the steps of:

determining whether a store instruction references a location in a local stack using an architecturally defined register for accessing a stack location;

allocating a rename register for the location, when the store instruction references the location using the architecturally defined register; and

replacing the reference to the location by a reference to the rename register.

27. The method according to claim 26, further comprising the step of inserting an operation, into an instruction stream containing the store instruction, to store the location from the rename register to a main memory of the system, upon replacing the reference to the location by the reference to the rename register.

28. The method according to claim 26, further comprising the step of determining whether the store instruction references a location in any stack, including the local stack, using another register, when the store

A2

instruction does not reference the location using the architecturally defined register.

29. The method according to claim 28, further comprising the step of performing the store instruction from one of a main memory and a cache of the system, when the store instruction does not reference the location using the other register.

30. The method according to claim 28, further comprising the step of executing a consistency-preserving mechanism to perform the store instruction to the stack area, when the store instruction references the location using the other register.

31. A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for renaming memory references to stack locations in a computer processing system, the method steps comprising:

detecting stack references that use architecturally defined stack access methods; and

A2

replacing the stack references with references to processor-internal registers.

32. The program storage device according to claim 31, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.

33. The program storage device according to claim 32, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

34. The program storage device program storage device according to claim 31, further comprising the step of performing a consistency-preserving operation for a stack reference that does not use the architecturally defined stack access methods.

35. The program storage device according to claim 34, wherein said step of performing a consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that

A2

references a stack area and that does not use the architecturally defined stack access methods.

36. The program storage device according to claim 34, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing a consistency-preserving operation comprises the step of recovering an in-order value for the stack reference from the main memory, upon performing said synchronizing step.

37. The program storage device according claim 36, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an instruction corresponding to the stack reference, when the stack reference is a write stack reference.

38. The program storage device according to claim 36, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

A2

39. The program storage device according to claim 34, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state of a processor the system;

synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.